一変数多項式因数分解のための並列計算系について(数式処理における理論と応用の研究)

AUTHOR(S):

藤瀬, 哲朗; 村尾, 裕一

CITATION:

藤瀬, 哲朗 ...[et al]. 一変数多項式因数分解のための並列計算系について(数式処理における理論と応用の研究). 数理解析研究所講究録 1998, 1038: 1-6

ISSUE DATE:

1998-04

URL:

http://hdl.handle.net/2433/61993

数理解析研究所講究録
1038巻 1998年 1-6

一変数多項式因数分解のための並列計算系について[1)]

藤瀬 哲朗 (三菱総合研究所)
村尾裕一 (東京大学大型計算機センター)

1. はじめに

スーパーコンピュータは，世間一般には科学技術計算における花形的存在と認知され，事実，これまでに多大な貢献をしてきている．しかしながら，多くの場合，それは数値計算におけることである．実際，スーパーコンピュータは，その性能を表す指標として Flop/s (FLoating OPerations per Second) が使われることが示すとおり，数値処理を主なターゲットとして開発されてきている．それに対して著者らはこのスーパーコンピュータや並列コンピュータを非数値処理である記号計算に適用する研究を続けているが, 本稿では並列処理, ここではタスク並列処理とデータ並列処理の双方の処理を利用したハイブリッド型の並列計算の研究である「統合並列処理技術の開発」([3]) の一環で開発している一変数因数分解の並列計算系について紹介する.

2. 並列計算に関する注意点

具体的な内容を挙げる前に，一般的な注意点を述べておこう.

ある計算処理に対し並列処理を施すには，先ず，その処理を構成する部分計算群の中に並行性 (concurrency) を見い出す必要がある．数式処理においては，具体的な処理内容の多くは数学的に定式化されており，例えば分配則が成り立つ演算の場合のように，その定式化の範囲内で，自明な並行性を見い出せる場合が数多くある．極端には，多項式の足し算における，各項の係数の計算という簡単な場合も考えられる．並行性が自明でない場合でも，処理を数理的に変形することによって，並行性を導き出しうる場合がある．例えば，多倍長整数を，中国剰余定理 (Chinese remainder theorem. 以下 CRT と略す) に基づいて，複数の法による剰余で表現するという方法は良く知られており，かつ，有用である．並列処理の研究においては，代数的独立性に基づく並行性を導出することが，その第一歩であり，それ自身で数理的に興味深い題材であることも多いが，それだけでは並列処理算法としては十分ではない．並列算法として設計するには，その部分計算群を，並行性と逐次性に基づき，並列計算機を構成する複数の演算要素の全体ができるだけ効率良く稼働するよ

[1)] 本研究開発の一部は, 情報処理振興事業協会 (IPA) が実施する創造的ソフトウェア育成事業の「統合並列処理技術の開発」として行われている.

うにマッピングすることが肝要である．この時，部分計算群の間で共用すべきデータの配置と流れや，その結果として部分計算をどの程度の粒度とすべきか等についても十分に考慮する必要がある．特に，数式処理の場合には，扱うデータには構造があり，かつ，その演算は複雑であるため，計算機ハードウェアで直接扱われる通常の数値計算の場合のように均質で等価であるとは，一般には期待できないことに注意する必要がある．例えば，数値データ一つとっても，数式処理では任意多倍長の整数として扱われるのが普通であるため，演算にかかるコストは決して一様であるとは期待できない．つまり，並行性が明らかであるからと言って，そのまま並列処理を実際に適用したとしても，全く功を奏さないということにもなりかねないのである．

3. 統合並列処理技術

統合並列処理技術研究の目的は，データ並列処理技術とタスク並列処理技術を統合する環境を開発することである．タスク並列では大きな粒度での処理負荷の割り当てを問題にするのに対し，データ並列はより小さな粒度での効率的並列実行が問題である．

具体的な目標は，統合並列処理環境構築のための言語機構および実装技術の開発であり，異なる並列処理機構間の同期機構および異なる並列処理機構間のデータ受渡し機構がポイントとなる．大上段に構えれば将来の並列計算機用 OS 技術の基礎となる技術の検証を目的としていると言っても良い．

データ並列処理とタスク並列処理との特徴を表 1 に示す．

Table 1. データ並列処理とタスク並列処理

	データ並列	タスク並列
計算の特徴	定型的	非定型的
計算量の予測	実行前に確定	実行時まで不確定
負荷の分散	実行前に決定	実行中に動的に
通信パターン	実行前に確定	実行時まで不確定
必要な通信	同期通信	非同期通信
自動並列化	実用段階 (並列化コンパイラなど)	研究途上 (一部を除き非効率)

データ並列処理とタスク並列処理は，

- **データ並列処理**
  - 規則的なデータ (密な行列など) に対する定型的並列計算技術
    ⟶ 限定的な適用範囲; 比較的容易な効率的実装
  - 現用の商用並列計算機環境下で効率的に処理可能
  - ここではベクタ計算も含める
- **タスク並列処理**

- 不規則なデータ, 非定型処理をのための並列計算技術
  ⟶ 広い適用範囲; 効率的実装は研究途上 (殊に細粒度時)
- 現在は実験環境での実現が中心; 同環境でのデータ並列は非効率
  現用の商用並列機ではソフトウェアサポートが貧弱

と考えられ, 結局商用並列機上のタスク並列処理環境の貧弱さとそれに由来するタスク並列の考え方でのデータ並列処理の非効率性が問題であると言える.

またタスク並列向きの問題もむりやりデータ並列で解いているのが現状である.

4. ターゲットとする計算機環境と実現方式

今回の実装のターゲットとなる計算機環境は以下の通りである.

ハードウェア: 並列型ベクタプロセッサ, 分散メモリ並列機,
共有メモリ並列機 (, 仮想共有メモリ並列機)

ソフトウェア: データ並列計算サポート (同期通信など)
タスク並列に必要な基本機構 (割り込みなど)
C 言語処理系

実現方式は次のとおり. タスク並列処理のベースとして並列論理型言語処理系 KLIC ([1]) を利用 (第五世代コンピュータ技術研究基盤化プロジェクトにおいて開発されたもの) する. この KLIC に, 対象システムの特徴に応じたデータ並列機構をプラグインし, 統合環境を形成する. もちろんハードウェアの差異の吸収 (共有メモリ, 分散メモリ, ベクタ型 など) やシステムソフトウェアの差異の吸収 (並列動作プロセサ群の見え方 など) のためにシステムごとに若干異なる実装技術の開発が必要となる. プログラミング言語の観点では, 非決定的計算, 非同期通信等のタスク並列制御部は KL1 言語に依り, その他は C 言語とデータ並列計算のためのパッケージに依る.

5. 一変数多項式因数分解のための並列計算系

一変数多項式因数分解の並列計算系の構築手段は次の通りである.

- まず問題の並行性を確認し, 次に並列化
- 高次の一変数多項式の因数分解技術を開発
- 基本的な処理に高速アルゴリズムを採用
- また多項式計算部分についてはベクタ計算化
- できるものは探索問題に持ち込み, タスク並列に帰着
- アルゴリズムは, 基本的に村尾と藤瀬の方法を採用

この中で並行性と並列性は次のようにとらえると良い. 論理的並列性はプログラムの正しい動作のために, どれとどれが並列に動いて良いかを示し, 物理的並列性はプログラムの効率的動作のために, どれとどれを並列に動かしたいか を示している. 単純に並行だから

並列にすれば良いのではなく, プロセサ有限個で通信コストがかかる現実の計算機では, 可能な並列実行すれば行なえば良いとは限らないのである.

効率的な並列実行に必要な留意事項は次の通りである.

- 負荷分散: なるべく多くのプロセサで処理
  → 全体の処理を早く終える
- 通信の削減: なるべく少ないプロセサで処理
  → 通信の量・頻度を抑える

部分問題間の通信量が少なく済むような問題の分割が必須であり, これは問題と解き方 (算法) に大きく依存する. すなわちこの留意事項は, 実は並列処理ソフトウェア研究の中心的課題とも言える.

6. 多項式因数分解算法の並行性と並列性

本計算における並行性と並列性を確認する.

- 全体の流れはパイプライン → 実行はスケジューリングの問題 (図 1)

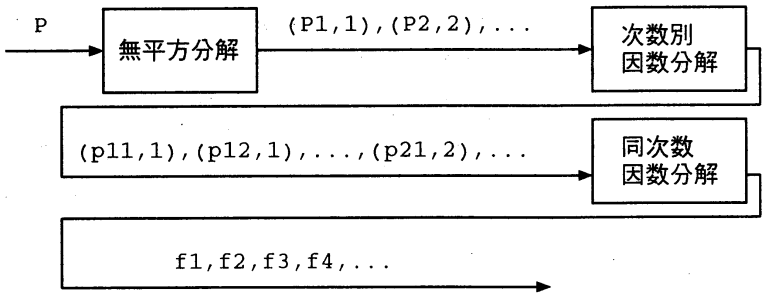

Fig. 1. 全体の流れ

- 並列 DDF → 村尾&藤瀬の ISSAC'96&PASCO'97 ([2],[5]
- 同次数因数分解
  - 線形因子の分離と因数分解 → 同位数因数分解 (図 2,[6])
  - それ以外は次数別因数分解と同じ手法
- 多項式基本 (四則) 演算 → 村尾の実験 ([8]). ベクトル計算を適用する.
- 多項式応用演算 (modular composition 等)
  Brent&Kung の算法あるいはその拡張版である Kaltofen&Shoup の算法を用い, 算法中の行列演算をベクトル計算する ([4]). SIMD 型計算となる.

現在開発中の計算系を図 3 に示す.

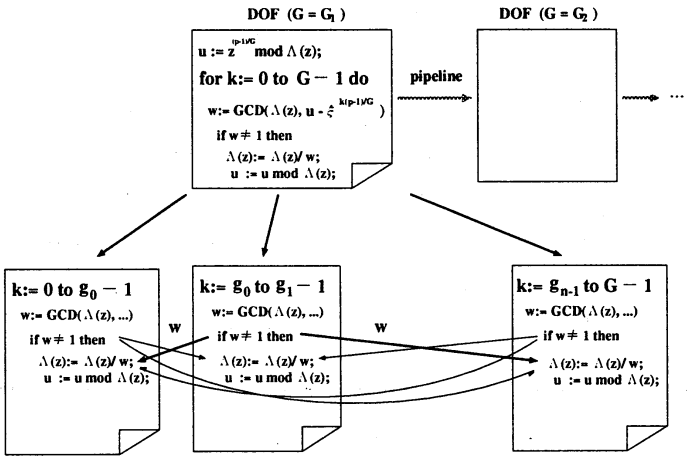

Fig. 2. DOF の並列化

7. 補足

今回の計算系は, 並列論理型言語処理系 KLIC で多項式のデータ並列 (ベクトル) 計算群を制御していると言っても良い. NTT 基礎研究所の平田氏によると KLIC での作法としては, プログラマの頭の中の処理イメージが

プログラム = アルゴリズム + データ構造
プログラム = 論理的な構造 + 制御

とすると, 並列処理の場合は, さらに

何を能動的存在 (プロセス) や受動的存在 (データ) とするか
どの物理的なプロセッサにどのプロセスとどのデータを 割り当てる のか

ということを考える必要があるとある. まさに今回の開発過程そのものである.

謝　辞

並列化に関する注意のうちいくつかは, 東京大学工学系研究科の近山隆教授のお話を参考にさせて頂いた. ここに記して謝意を表したい.

参 考 文 献

[1] Chikayama, T., Fujise, T., and Sekita, D.: A portable and efficient implementation of KL1, *Programming Language Implementation and Logic Programming, Proc. 6th PLILP '94*, *LNCS*, 844, Springer-Verlag, pp.25–39 (1994).

[2] Fujise, T. and Murao, H.: Parallel distinct degree factorization algorithm, *Proccedings of ISSAC96*, pp.18–25 (1996).

[3] 藤瀬, 近山: 統合並列処理技術の開発, 創造的ソフトウェア育成事業及びエレクトリック・コマース推進事業中間成果発表会論文集「創造的ソフトウェア育成事業編」, 情報処理振興事業協会, pp.117–122 (1997).

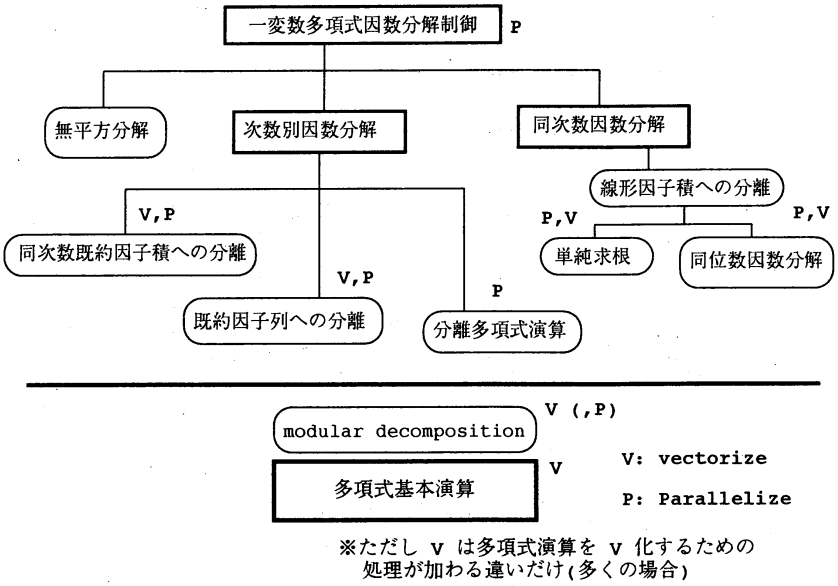

Fig. 3. 並列一変数多項式因数分解計算系

[4] Kaltofen, E. and Shoup, V.: Subquadratic-time factoring of polynomials over finite fields, *Proc. 27th Annual ACM Symposium on the Theory of Computing*, Las Vegas, Nevada, pp.398–406 (1995).

[5] Murao, H. and Fujise, T.: Towards an efficient implementation of a fast algorithm for multi-point polynomial evaluation and its parallel processing, *Proceedings of PASCO'97*, pp.24–30 (1997).

[6] Murao, H. and Fujise, T.: Modular algorithm for sparse multivariate polynomial interpolation and its parallel implementation, *J. Symb. Comput.*, **22**, pp.1–22 (1996).

[7] 村尾: $\mathbf{Z}_p$ 上の多項式の因数分解 — 高速化技法・ベクトル処理・並列処理—, 講究録 920 「数式処理における理論とその応用の研究」, 京都大学数理解析研究所, 1995.

[8] 村尾: 高速多項式乗算アルゴリズムの実現, 日本数式処理学会第 6 回大会, 1997.